Understanding Quantum Software Engineering Challenges
An Empirical Study on Stack Exchange Forums and GitHub Issues

Mohamed Raed El aoun, Heng Li, Foutse Khomh, Moses Openja
Department of Computer Engineering and Software Engineering
Polytechnique Montréal, Montréal, QC, Canada
{mohamed-raed.el-aoun, heng.li, foutse.khomh, moses.openja}@polymtl.ca

***Abstract*—With the advance of quantum computing, quantum software becomes critical for exploring the full potential of quantum computing systems. Recently, quantum software engineering (QSE) becomes an emerging area attracting more and more attention. However, it is not clear what are the challenges and opportunities of quantum computing facing the software engineering community. This work aims to understand the QSE-related challenges perceived by developers. We perform an empirical study on Stack Exchange forums where developers post-QSE-related questions & answers and Github issue reports where developers raise QSE-related issues in practical quantum computing projects. Based on an existing taxonomy of question types on Stack Overflow, we first perform a qualitative analysis of the types of QSE-related questions asked on Stack Exchange forums. We then use automated topic modeling to uncover the topics in QSE-related Stack Exchange posts and GitHub issue reports. Our study highlights some particularly challenging areas of QSE that are different from that of traditional software engineering, such as explaining the theory behind quantum computing code, interpreting quantum program outputs, and bridging the knowledge gap between quantum computing and classical computing, as well as their associated opportunities.**

***Index Terms*—Quantum computing, Quantum software engineering, Topic modeling, Stack Exchange, Issue reports.**

I. Introduction

Over the past decades, quantum computing has made steady and remarkable progress [1]–[3]. For example, IBM Quantum [4] now supports developers to develop quantum applications using its programming framework and execute them on its cloud-based quantum computers. Based on the quantum mechanics principles of `superposition` (quantum objects can be in different states at the same time) [5] and `entanglement` (quantum objects can be deeply connected without direct physical interaction) [6], quantum computers are expected to make revolutionary computation improvement over today's classical computers [7]. In particular, quantum computing is expected to help solve the computational problems that are difficult for today's classical computers, including problems in cryptography, chemistry, financial services, medicine, and national security [8].

The success of quantum computing will not be accomplished without quantum software. Several quantum programming languages (e.g., QCL [9]) and development tools (e.g., Qiskit [10] have been developed since the first quantum computers. Large software companies like Google [11], IBM [4], and Microsoft [12] have developed their technologies for quantum software development. Quantum software developers have also achieved some preliminary success in applying quantum software to certain computational areas (e.g, machine learning [13], optimization [14], cryptography [15], and chemistry [16]). However, there still lacks large-scale quantum software. Much like Software Engineering is needed for developing large-scale traditional software, the concept of Quantum Software Engineering (QSE) has been proposed to support and guide the development of large-scale, industrial-level quantum software applications. This concept has been gaining more and more attention recently [3], [8], [17]. QSE aims to apply or adapt existing software engineering processes, methods, techniques, practices, and principles to the development of quantum software applications, or create new ones [8]. Pioneering work sheds light on new directions for QSE, such as quantum software processes & methodologies [18], quantum software modeling [19], and design of quantum hybrid systems [20]. In the meanwhile, we observe an exponential increase of discussions related to quantum software development on technical Q&A forums such as Stack Overflow(e.g. from 8 in 2010 to 1434 in 2020). We also notice an increasing number of quantum software projects hosted on GitHub, where developers use issue reports to track their development and issue fixing processes. Such technical Q&As and issue reports may communicate developers' faced challenges when developing quantum software applications.

In this paper, we aim to understand the challenges perceived by quantum software developers and seek opportunities for future QSE research and practice. In particular, we examine technical Q&A forums where developers ask QSE-related questions, and GitHub issue reports where developers raise QSE-related issues. We apply a series of heuristics to search and filter Q&A posts that are related to QSE and to search and filter GitHub projects that are related to quantum software. In total, we extract and analyze 3,117 Q&A posts and 43,979 Github issues that are related to QSE. We combine manual analysis and automated topic modeling to examine these Q&A posts and Github issues, to understand the QSE challenges developers are facing. In particular, our study aims to answer the three following research questions (RQs):

RQ1: *What types of QSE questions are asked on technical forums?* To understand the intention behind developers' questions on technical forums and the types of information that they are seeking, we manually examined a statistically representative sample of questions. We extended a previous taxonomy from prior work [21] and found nine categories of questions. Our results highlight the need for future efforts to support developers' quantum program development, in particular, to develop learning resources, to help developers fix errors, and to explain the theory behind quantum computing code.

RQ2: *What QSE topics are raised in technical forums?* The QSE-related posts may reflect developers' challenges when learning or developing quantum programs. To understand their faced challenges, we use topic models to extract the semantic topics in their posts. We derived nine topics including traditional software engineering topics (e.g., `environment management` and `dependency management`) and QSE-specific topics (e.g., `quantum execution results` and `quantum vs. classical computing`). We highlighted some particularly challenging areas for QSE, such as interpreting quantum program outputs, understanding quantum algorithm complexity, and bridging the knowledge gap between quantum computing and classical computing.

RQ3: *What QSE topics are raised in the issue reports of quantum-computing projects?* Issue reports of quantum computing projects record developers' concerns and discussions when developing these projects. Thus, we analyze the topics in the issue reports to understand the challenges are developers facing in practical quantum computing projects. We observe that the QSE-related challenges that we derived from forum posts indeed impact practical quantum program development in these GitHub projects, while GitHub issues bring new perspectives on developers' faced challenges (e.g., on specific quantum computing applications such as machine learning). We also observe that such challenges are general among quantum computing projects.

Paper organization. The rest of the paper is organized as follows. In Section II we discuss the background about quantum software engineering and the related work. Then, in Section III we describe the design of our study. In Section IV we present our results. Section V discusses threats to the validity of our findings. Finally, Section VI concludes the paper.

II. Background and Related Work

This study aims to understand the quantum software engineering challenges through examining technical forum posts and GitHub issue reports. In this section, we present the background and prior work related to our study. First, we describe the background and related work of quantum computing, quantum programming, and quantum software engineering. Then, we discuss prior work that performs topic analysis on technical forum posts and issue reports.

A. Quantum Computing

Quantum computers aim to leverage the principles of quantum mechanics such as `superposition` and `entanglement` to provide computing speed faster than today's classical computers. While classical computers use bits in the form of electrical pulses to represent 1s and 0s, quantum computers use quantum bits or **Qubits** in the form of subatomic particles such as electrons or photons to represent 1s and 0s. A Qubit, unlike a classical bit, can be 0 or 1 with a certain probability, which is known as the **superposition** principle [22]. In other words, a quantum computer consisting of Qubits is in many different states at the same time. When a Qubit is **measured**, it collapses into a deterministic classical state. The status of two or more of Qubits can be correlated (or entangled) in the sense that changing the status of one Qubit will change the status of the others in a predictable way, which is known as the **entanglement** phenomenon [22]. The `superposition` and `entanglement` phenomenons give quantum computers advantages over classical computers in performing large-scale parallel computation [22].

Similar to classical logic gates (e.g., `AND`, `OR`, `NOT`), **quantum logic gates** (or **quantum gates**) alter the states (the probability of being 0 or 1) of the input Qubits. Like classical digit circuits, **quantum circuits** are collections of quantum logic gates interconnected by quantum wires. Figure 1 illustrates the architecture of a quantum computer [3], [23]. The architecture contains two layers: a quantum computing layer where the quantum physics and circuits reside, and a classical computing layer where the quantum programming environment and software applications reside.

- *Physical building blocks*: physical realization of Qubits and their coupling/interconnect circuitry.
- *Quantum logic gates*: physical circuitry for quantum logic gates.
- *Quantum-classical computer interface*: the hardware and software that provides the boundary between classical computers and the quantum computing layer.
- *Quantum programming environment*: quantum programming languages and development environment.
- *Business applications*: quantum software applications (based on quantum programming languages) that meet specific business requirements

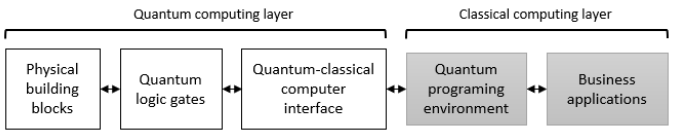
Quantum computing layer
Classical computing layer
Physical building blocks
Quantum logic gates
Quantum-classical computer interface
Quantum programing environment
Business applications

Fig. 1. The architecture of a quantum computer [3], [23]

B. Quantum Programming

Quantum computing as a new general paradigm can massively influence how software is developed [3], [8], [24]. Quantum programming is the process to design an executable quantum program to accomplish a specific task [24]. Quantum programming uses syntax-based notations to represent and operate quantum circuits and gates. Early efforts of quantum programming language development focused on the quantum Turing machine [25] but did not produce practical quantum programming languages. Later efforts have turned to the quantum circuits model where the quantum system is controlled by a classical computer [26]. This concept has given birth to many new quantum programming languages such as qGCL [27], LanQ [28], Q# [29] and Qiskit [10]. Prior work conducted extensive exploration along the lines of quantum programming [24] and quantum software development environments [30]. The survey [3] also provides a comprehensive overview of research works along these lines.

C. Quantum Software Engineering

Quantum software engineering (QSE) is still in its infancy. As the result of the first International Workshop on Quantum Software Engineering & Programming (QANSWER), researchers and practitioners proposed the "Talavera Manifesto" for quantum software engineering and programming, which defines a set of principles about QSE [8], including: *(1) QSE is agnostic regarding quantum programming languages and technologies; (2) QSE embraces the coexistence of classical and quantum computing; (3) QSE supports the management of quantum software development projects; (4) QSE considers the evolution of quantum software; (5) QSE aims at delivering quantum programs with desirable zero defects; (6) QSE assures the quality of quantum software; (7) QSE promotes quantum software reuse; (8) QSE addresses security and privacy by design; and (9) QSE covers the governance and management of software.*

Zhao [3] performed a comprehensive survey of the existing technology in various phases of quantum software life cycle, including requirement analysis, design, implementation, testing, and maintenance. Prior work [17]–[20] also discussed challenges and potential directions in QSE research, such as modeling [19] and quantum software processes & methodologies [18], and design of quantum hybrid systems [20]. Different from prior work, this work makes the first attempt to understand the challenges of QSE perceived by practitioners.

D. Topic Analysis of Technical Q&As

Prior work performs rich studies on technical Q&A data, especially on Stack Exchange data [31]. Here we focus on prior work that performs topic analysis on technical Q&A data. Topic models are used extensively in prior work to understand the topics of general Stack Overflow posts and the topic trends [19], [32]–[34]. Prior work also leverages topic models to understand the topics of Stack Overflow posts related to specific application development domains, such as mobile application development [35], [36], client application development [37], machine learning application development [38], as well as concurrency [39] and security [40] related development. In addition, prior work leverages topic models to understand non-functional requirements communicated in Stack Overflow posts [41], [42]. Zhang et al. [43] use topic models to detect duplicate questions in Stack Overflow. Finally, Treude et al. [44] proposes an automated approach to suggest configurations of topic models for Stack Overflow data. Most of these studies use the Latent Dirichlet Allocation (LDA) algorithm or its variants to extract topics from the technical Q&A data. In this work, we also leverage the widely used LDA algorithm to extract topics from the technical Q&A data related to quantum software enginering.

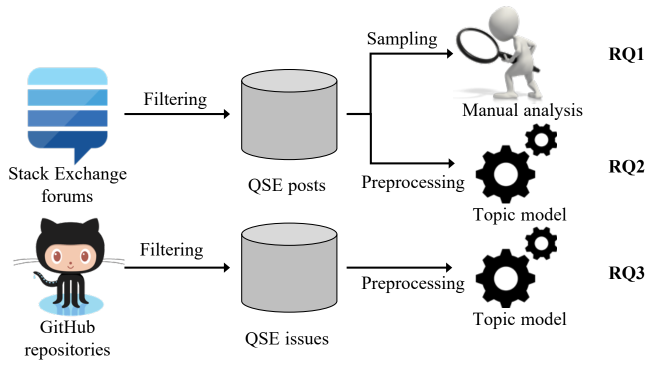

Fig. 2. Overview of our empirical study

E. Topic Analysis of Issue Reports

Issue reports have been widely explored in prior work. Here we focus on studies that apply topic analysis on issue report data. Prior work leverages topic models to automatically assign issue reports to developers (*a.k.a.* bug triage) [45]–[48]. These studies first uses topic models to categorize the textual information in the issue reports, then learn mappings between the categorized textual information and developers. Prior work also leverages topic models to automatically detect duplicate issue reports based on the similarity of their topics [49]–[51]. Nguyen et al. [52] use topic models to associate issue reports and source code based on their similarities, in order to help developers narrow down the searched source code space when resolving an issue. Finally, prior work also studies the trends of topics in issue reports [53], [54]. LDA and its variants are the most popular topic modeling approaches used in these studies. Therefore, we also leverage LDA to extract topics from GitHub issue reports related to QSE.

III. Experiment Setup

This section describes the design of our empirical study.

A. Overview

Figure 2 provides an overview of our empirical study. We study QSE-related posts on Stack Exchange (SE) forums and the issue reports of quantum computing GitHub projects. From Stack Exchange forums, we first use tags to filter QSE-related posts. In RQ1, we manually analyze a statistically representative sample of these posts to understand the type of information sought by developers. In RQ2, we use automated topic models to analyze the topics of these posts and their characteristics. From GitHub repositories, we first apply a set of heuristic rules to filter the quantum computing projects. Then we extract the issue reports of these quantum computing projects. Finally, we perform topics modeling on these issue reports to analyze the topics in the textual information of the issue reports (RQ3). We describe the details of our data collection and analysis approaches in the rest of this section.

B. Stack Exchange forums data collection

We follow three steps to collect QSE related data from Stack Exchange forums. First, we collect Q&A data from four Stack Exchange forums. Second, we identify a set of tags that are related to QSE. Finally, we use the identified tags to select the posts that are related to QSE. We explain the steps below.

Step 1: Collecting technical Q&A data. We extract technical Q&A data from four Stack Exchange forums: Stack Overflow [55], Quantum Computing Stack Exchange [56], Computer Science Stack Exchange [57], and Artificial Intelligence Stack Exchange [58]. We consider the Stack Overflow forum as it contains posts related to quantum programming and it is widely used for studying various software engineering topics (e.g., mobile app development [35], [36], machine learning application development [38], etc.). We consider the other three forums because they contain posts that discuss topics related to quantum computing and quantum programming. We extracted the post data from these forums with the help of the Stack Exchange Data Explorer [59]. Stack Exchange data explorer holds an up to date data for these forum between 08-2008 and 03-2021.

Step 2: Identifying tags related to QSE. The studied Stack Exchange forums use user defined tags to categorize questions. We follow two sub-steps to select the tags that are related to QSE. We started by searching for questions with the tag "quantum-computing" in the entire Stack Exchange dataset D through the data exchange explorer. We obtained 254 questions tagged with "quantum-computing" from the studied forums. After manually inspecting the 30 most voted questions, we selected an initial tag set T_{init} consisting of ten tags including "quantum-computing", "qiskit", "qsharp", "q#", "quantum-development", "quantum-circuit", "ibmq", "quantum-ai", "qubit" and "qutip". Then we extracted the questions related to T_{init} from the initial dataset D and obtained a new set of questions P. In order to expand the initial tag set, we extracted the frequently co-occurring tags with T_{init} from P and build a new tag set T_2.

Not all the tags in T_2 are related to quantum computing. To determine the final tag set T_{final}, following previous work [60] [61], we filter the tags in T_2 based on their relationships when the initial tag set T_{init}. For each tag t in T_2, we calculate:

$$(\text{Significance})\, \alpha(t) = \frac{\#\text{ of questions with tag } t \text{ in } P}{\#\text{ of questions with tag } t \text{ in } D} \qquad (1)$$

$$(\text{Relevance})\, \beta(t) = \frac{\#\text{ of questions with tag } t \text{ in } P}{\#\text{ of questions in } P} \qquad (2)$$

To select a tag t, the value of significance-relevance $\alpha(t)$, $\beta(t)$ need to be higher than a threshold we set. To select the optimal threshold values for α and β, we experimented with a set of values respectively between 0.05, 0.35 and 0.001, 0.03. For each α and β and for each tag above the threshold, we inspected the top 10 most voted posts and verified if the tag is related to QSE, we ended up with the optimal threshold respectively equal to 0.005 and 0.2 which are consistent with previous work [62] [61]. The final tag set T_{final} is formed of 37 tags in total. Since quantum computing is a wide topic and our focus is QSE, we further manually inspected the description of each tag t in T_{final} and the top 10 questions of each tag in each studied forum to remove tags that are not related to QSE. Finally our tag set T_{final} was reduced from 37 to 18 tags (14 unique tags as different forums have tags with the same names). Table I lists our final set of tags.

TABLE I
OUR SELECTED TAGS AND THE NUMBER OF QUESTIONS AND ANSWERS

Stack Ex. forum	Tag set	#Q	#A
Stack overflow	post-quantum-cryptography, q#, quantum-computing, qiskit, qcl, qutip, qubit, tensorflow-quantum	250	183
Quantum computing	programming, classicalcomputing, q#, qiskit, cirq, ibm-q-experience, machine-learning, qutip	1534	778
Computer science	quantum-computing	238	117
Artificial intelligence	quantum-computing	13	4

Step 3: Selecting questions and answers. We extract the final sets of questions and answers using the final tag sets shown in Table I. We select all the posts that are tagged with at least one of the tags. We ended up with a total of 3,117 questions and answers from the four considered forums in our data set D_{final}. 35% of the final data are answers where 65% are questions. The number of posts (questions and answers) extracted from each forum is shown in Table I.

C. GitHub issues data collection

In this work, we study the issue reports of quantum computing projects on GitHub. We downloaded the GitHub selected quantum computing projects issues in March 2021. We follow three steps described below to extract the issue reports of quantum computing projects from GitHub.

Step 1: Searching candidate projects. We search for quantum computing related projects using three criteria: 1) The description of the project must be in English (i.e., for us to better understand the content). 2) The project name or description must contain the word "quantum" (the word quantum is case sensitive in the project name or description). 3) The project is in a mainline repository (i.e., not a fork of another repository). We end up with a total of 1,364 repositories.

Step 2: Filtering quantum computing projects. We filter the searching result and identify quantum computing related projects with three criteria: 1) To avoid selecting student assignments, following previous work [63] [64], we select repositories that were forked at least two times. 2) The projects must have a sufficient history of development for us to analyze the issue reports. Therefore, we select the projects that were created at least 10 months earlier than the data extraction date. Moreover, only the projects that have at least 100 commits and 10 issues are selected. 3) To ensure the quality of the project selected, we manually inspect the projects' descriptions and remove projects that are not related to quantum computing, projects that are created for hosting quantum computing related documentation, as well as lecture notes related to quantum computing. Finally, we obtain a total of 122 projects directly related to quantum computing applications.

Step 3: Extracting issue reports. We use the GitHub Rest API [65] to extract all the issue reports of the final 122 projects on GitHub. In total, we obtain 43,979 issue reports.

D. Data pre-processing for topic modeling

We build one topic model on the Stack Exchange forum data and another topic model on the GitHub issue data. Below we describe how we pre-process these two types of data before feeding them into topic models.

Pre-processing Q&A post data. We treat each post (i.e., a question or an answer) as an individual document in the topic model. For each question, we join the title and the body of the question to create a single document. As Q&A posts contain code snippets between <code> and </code> which may bring noise to our topic models, we remove all text between <code> and </code>. We also remove HTML tags (e.g., <p></p>), URLs and images from each post. In addition, we remove stop words (e.g., "like", "this", "the"), punctuation, and non-alphabetical characters using the Mallet and NLTK stop words set. Finally, we apply the Porter stemming [66] to normalize the words into their base forms (e.g., "computing" is transformed to "comput"), which can reduce the dimensionality of the word space and improve the performance of topic models [67]

Pre-processing issue report data. We treat each issue report as an individual document in the topic model. We join the title and the body of each issue as a single document. Similarly, we remove code snippets, URLs and images from the issue body. Since there are no tags in GitHub issues that identify code snippets, we look for backquote " " or triple backticks "' in the content of the issues and remove the code enclosed between this punctuation. We also remove stop words, non-alphabetical characters, and punctuation. Finally, we apply Porter stemming to normalize the words into their base forms.

E. Topic modeling

We use automated topic modeling to analyze the topics in the Q&A posts and issue reports. Specifically, we use the Latent Dirichlet Allocation (LDA) algorithm [68] to extract the topics from both of our datasets. LDA is a probabilistic topic modeling technique that derives the probability distribution of frequently co-occurred word sets (i.e., topics) in a text corpus. A topic is represented by a probability distribution of a set of words, while a document is represented as a probability distribution of a set of topics. LDA is widely used for modeling topics in software repositories [69], including technical Q&A posts (e.g, [70]) and issue reports (e.g., [49]). We use two separate topic models to extract the topics from the Q&A post data and the issue report data. For a better performance of the topic modeling and a good classification quality, following previous work [61] [71], we consider both uni-gram and bi-gram of words in our topic models.

LDA Implementation. We use the Python implementation of the Mallet topic modeling package [72] to perform our topic modeling. The Mallet package implements the Gibbs sampling LDA algorithm and uses efficient hyper-parameter optimization to improve the quality of the derived topics [72].

Determining topic modeling parameters. The number of topics (K) is usually manually set by the user as it controls the granularity of the topics [61]. The α parameter controls the topic distribution in the documents (i.e., Q&A posts or issue reports), while the β parameter controls the word distribution in the topics. In this work, we use the topic coherence score [73] to evaluate the quality of the resulting topics and determine the appropriate parameters (K, α, and β), similar to prior work [29], [61]. The coherence score measures the quality of a topic by measuring the semantic similarity between the top words in the topic. Thus, this score distinguishes between topics that are semantically interpretable and topics that are coincidences of statistical inference [73]. Specifically, we use the Gensim Python package's `CoherenceModel` [74] module to calculate the coherence scores of the resulting topics. To capture a wide range of parameters and keep the topics distinct from each other, we experiment with different combination of the parameters, by varying the values of K from 5 to 30 incremented by 1 each time, the values of document-topic distribution α from 0.01 to 1 incremented by 0.01 [75], and the values of word-topic distribution β from 0.01 to 1 incremented by 0.01 [75]. We retain the resulting topics with the highest average coherence score.

After getting the automatically derived topics, we manually analyze the resulting topics and assign meaningful labels to the topics. We elaborate more on this process in RQ2 and RQ3 for the Q&A post topics and the issue report topics, respectively.

IV. Experiment Results

In this section we report and discuss the results of our three research questions. For each research question, we first present the motivation and approach, then discuss the results for answering the research question.

RQ1: What types of QSE questions are asked on technical forums?

*1) **Motivation**:* In order to understand QSE challenges developers are facing, we first want to understand what types of questions they are asking (e.g., whether they are asking questions about using APIs or fixing errors). This is important to identify the areas in which QSE developers should be supported and the type of resources that they need. Similar to prior work [76], we focus on the intent behind the questions asked by QSE developers instead of the topics of the questions.

*2) **Approach**:* To identify the type of questions that users are asking in technical forums, we performed a manual analysis of a statistically representative sample from our studied QSE questions. We sampled 323 questions with a confidence level of 95% and a confidence interval of 5%. For each question, we examined its title and body, to understand the intent of the user who posted the question. We used a hybrid card sorting approach to perform the manual analysis and assign labels (i.e., types of questions) to each sampled question. Specifically, we based our manual analysis on an existing taxonomy of the types of questions asked on Stack Overflow [76] and added new types when needed. For each question we assigned one label; in case a question is associated with two or more labels, which we found only in a few cases, we chose the most relevant one.

Hybrid card sorting process. Two authors of the paper (i.e., coders) jointly performed the hybrid card sorting. We split the sampled data into two equal subsets and performed the sorting in two rounds, similar to prior work [77]. Our process guaranteed that each question is labelled by both coders.

1) **First-round labeling.** Each coder labels a different half of the questions independently.
2) **First-round discussion.** In order to have a consistent labeling strategy, we had a meeting to discuss the labeling results in the first round and reached an agreed-upon set of labels. A third author of the paper is involved in the discussion.
3) **Revising first-round labels.** Each coder updated the first round labeling results based on the discussion.
4) **Second-round labeling.** Each coder labeled the other half of the questions independently based on the agreed-upon labels in the first round. New labels are allowed in this round.
5) **Second-round discussion.** We had a meeting to discuss the second-round labeling results, validate newly added labels and verify the consistency of our labels. A third author is also involved in the discussion.
6) **Revising second-round labels.** Based on the second-round discussion, each coder revised the labels and finalized its individual labeling of the questions. We calculate the inter-coder agreement after this step.
7) **Resolving disagreement.** We had a final meeting to resolve the disagreement in our labeling results and reached the final label for each question. For each difference in our labels, the two coders and a third author discussed the conflict and reached a consensus.

Inter-coder agreement. We measured the inter-coder agreement between the coders and obtained a Cohen's kappa k value of 0.73 which indicates a substantial agreement [78]. Therefore our manual labeling results are reliable.

*3) **Results**:* Table II shows the result of our qualitative analysis for identifying the categories of questions in technical forums. Among the 323 questions we analyzed, we could not assign a label to only one question. In the table, we provide the description of each category and how frequent it appears in our qualitative analysis.

All seven categories of Stack Overflow questions identified in prior work appear in QSE-related posts. Prior work [76] identified seven categories of questions on Stack Overflow by studying Android-related questions, including `API usage`, `Conceptual`, `Discrepancy`, `Errors`, `Review`, `API change`, and `Learning`, ordered by their occurrence frequency. Although quantum computing is still a new area, people start to ask all these different categories of questions, indicating that quantum computing face similar software engineering challenges (e.g., `API usage` and `API change`) as other software engineering domains. Similar to prior work, we find that `API usage` is the most frequent category with 26.3% instances. The questions of this category are usually identified by "how to"; e.g., "*How to return measurement probabilities from the QDK Full-state simulator?*"

The categories of `Errors` and `Learning` are relatively more frequent in QSE-related questions than in the prior taxonomy of question categories [76]. Compare to prior work [76] on classifying Android-related questions, we find that `Errors` and `Learning` questions are relatively more frequent. As quantum computing is still an emerging domain, people practicing it face many errors when developing quantum computing applications and they find it challenging to find learning resource for quantum computing. An example of the `Errors` category is "*I have Qiskit installed via Anaconda and a virtual environment set up in Python 3.8. ... I get an error. I'm not sure what the problem is. How do I fix it?*". Another example for the `Learning` category is "*How do I learn Q#? What languages should I know prior to learning Q#? How do I get started with quantum computing?*". These findings suggest the need to develop tools or resources to help developers avoid or address such errors, as well as developing tutorials, books, and other learning resources to help beginners get acquainted with quantum computing.

Two new categories of questions (i.e., `Theoretical` and `Tooling`) emerge in QSE-related posts. In fact, the category of `Theoretical` is the second most frequent among all categories. This category is usually associated with keywords such as "can someone explain", "what is", and "does quantum". An example question of this category is "What is the analysis of the Bell Inequality protocol in Cirq's 'examples'?" where Cirq [79] is a Python library for developing quantum computing applications. This category of questions indicates that people have challenges understanding the theoretical concepts behind quantum computing code. Future efforts are needed to explain such theoretical concepts for developers. The category of `tooling` represents questions that are looking for tools, frameworks, or libraries that can help solve a QSE-related problem or verifying whether a tool, framework, or library can help solve a problem. For example, "*I want to use Blender and Blender Python Scripts working with Qiskit. How can I do this? How to make communication between Blender and Qiskit installed with Anaconda Python?*". This category indicates the lack of established tools for supporting quantum program development.

We identified nine categories of QSE-related questions in Stack Exchange forums. The categories `Theoretical`, `Errors`, `Learning`, and `Tooling` are new or become more frequent in QSE-related questions. Our results highlight the need for future efforts to support developers' quantum program development, in particular, to develop learning resources, to help developers fix errors, and to explain theory behind quantum computing code.

TABLE II
A TAXONOMY OF QUESTION CATEGORIES WHICH BASES ON AND EXTENDS [76]

Category	Description	Freq
API usage	Questions of this category are usually identified by "how to", i.e., how to use an API or how to implement a functionality.	85
Theoretical*	This category of questioners ask about theoretical explanations of quantum programs, algorithms, and concepts.	54
Errors	This category of questions search for explanations and solutions of errors and exceptions when developing or executing quantum programs.	49
Conceptual	Questions in this category are related to the limitation, background and the underlying concept of an API.	45
Discrepancy	Question of this category usually ask for explanations or solutions for unexpected results (e.g., "what is the problem", "why not work".	31
Learning	Questions in this category are searching for learning resources such as documentation, research papers, tutorials, or websites.	22
Review	This category describes questions like: "How/Why this is working?" or "Is there a better solution?". Generally, the questions in this category look for a better solution to a problem or for help reviewing the current solution.	17
Tooling*	This category describes questions like "I am looking for ...", "Is there a tool for ...". These questions search for tools to solve a specific problem or check the features of a tool.	16
API change	This category of questions concern about changes of an API and the associated compatibility issues and other implications.	2

*Categories newly identified in QSE-related questions.

RQ2: What QSE topics are raised in technical forums?

*1) **Motivation**:* Developers post QSE-related questions and answers on technical forums. Their posts may reflect their faced challenges when learning or developing quantum programs. To understand their faced challenges, we use topic models to extract the semantic topics in their posts and analyze the characteristics of these topics.

*2) **Approach**:* **Topic assignment and frequency.** The automated topic modeling generated nine topics and distribution of co-occurring words in each topic. We then manually assigned a meaningful label to each topic. Following prior work [61], [80], [81], to assign a meaningful label to a topic, the first author first proposed labels using two pieces of information: (1) the topic's top 20 keywords, and (2) the top 10-15 most relevant questions associated with the topic. Then, three authors of the paper reviewed the labels in meetings and reassigned the labels when needed. We obtained a meaningful label for each of the nine topics at the end. For each topic, we measure the percentage of the posts (i.e., frequency) that have it as the dominant topic (i.e., with the highest probability).

Topic popularity. To understand developers' attention towards each topic, following previous work [61], [80], [81], we measured three metrics for each topic: (1) the median number of views of the associated posts, (2) the median number of associated posts marked as `favorite`, and (3) the median score of the associated posts. For each topic, the associated posts refer to the posts that have it as the dominant topic.

Topic difficulty. In order to better understand the most challenging aspects for developers, we measure the difficulty of each topic in terms of how difficult it is for the associated posts to get accepted answers. Following prior work [61], [80], [81], for each topic, we measure two metrics: (1) the percentage of the associated questions with no accepted answer, and (2) the median time required by the associated questions to get an accepted answer (only considering the ones with an accepted answer). For each topic, the associated questions refer to the questions that have the topic as the dominant topic.

*3) **Results**:* **We derived nine topics that are discussed in QSE-related posts, including traditional software engineering topics (e.g., `environment management` and `dependency management`) and QSE-specific topics (e.g., `quantum execution results` and `Quantum circuits`)**. Table III describes the nine topics and their frequency in the analyzed posts. Table IV shows the median views, scores, and favorites of the posts associated with these topics. The three most dominant topics are `environment management`, `dependency management`, and `algorithm complexity`.

`Environment management` is the most dominant topic representing 15.03% of posts. For example, the most viewed question of this topic is "*I downloaded the Quipper package but I have not been able to get haskell to recognize where all of the modules and files are and how to properly link everything*" which gained 2772 views. Other examples include "How can I run QCL (quantum programming language) on Windows?" and "Visualization of Quantum Circuits when using IBM QISKit". We can observe that users are new to quantum computing and facing problem while setting up their environment and installing their tools. This topic is also linked to the question category `tooling` that we derived in RQ1.

`Dependency management` is the second most discussed topic representing 14.82% of the posts. For example, the most viewed question (with 2,239 views) of this topic is "*When trying the above code, I am receiving the following error: ModuleNotFoundError: No module named qiskit*" where `qiskit` is an open source framework for quantum program development [10]. We noticed that a large number of questions are directly related to `qiskit`. This can be explained by the lack of documentation or tutorials in using this framework.

`Algorithm complexity` is the third most dominant topic. This topic is about understanding the complexity of quantum algorithms and how to optimize quantum algorithms. For example, the most viewed question of this topic is "*For the other algorithms, I was able to find specific equations to calculate the number of instructions of the algorithm for a given input size (from which I could calculate the time required to calculate on a machine with a given speed). However, for Shor's algorithm, the most I can find is its complexity: O($(log\ N)^3$)*", which receives 4,718 views. This topic is linked to the questions category `theoretical` derived from RQ1. This topic indicates developers' challenge in understanding the complexity of quantum algorithms.

TABLE III

TOPICS EXTRACTED FROM QSE RELATED POSTS ON STACK EXCHANGE FORUMS

Topic (manual label)	**Keywords**	**Description**	**% Freq**
Environment management	quot, error, python, build, code	Development environment and build problems	15.03
Dependency management	qiskit, import, ibmq, operator, provider	Library installation, use, and versioning issues	14.82
Algorithm complexity	time, problem, algorithm, number, function	Quantum algorithm complexity and optimization	14.06
Quantum execution results	circuit, result, back-end, simulator, measure	Quantum program execution results on quantum backends (e.g., simulators)	13.22
Learning resources	question, paper, work, understand, answer	Searching for learning resources such as research papers and tutorials	9.05
Data structures and operations	matrix, return, array, datum, list	Data structures (e.g., matrix, arrays and list) and their operations in quantum programs	8.81
Quantum circuits	qubit, gate, control, operation, cirq	Elements of quantum circuits (e.g., Qubits, gates) and their operations	8.66
Quantum vs. classical computing	quantum, computer, classical, computing, algorithm	Comparisons between quantum and classical computing or migrating classic algorithms to quantum computing	8.30
Quantum algorithms understanding	state, rangle, frac, theta, sqrt	Quantum algorithm explanation and interpretation	7.51

TABLE IV

POPULARITY OF QSE-RELATED TOPICS ON STACK EXCHANGE FORUMS

Topic name	$\tilde{View}$	$\tilde{Score}$	$\tilde{Favorite}$
Quantum vs. classical computing	147.5	3	1.5
Quantum circuits	107.0	2	1.0
Environment management	106.0	1	1.0
Learning resources	102.0	2	1.0
Quantum execution results	98.0	1	1.0
Quantum algorithms understanding	97.5	2	1.0
Dependency management	93.0	2	1.0
Algorithm compolexity	87.5	1	1.0
Data structures and operations	82.0	1	1.0

As quantum programming is oriented to searching solutions in a probabilistic space, which is counter-intuitive from the classical computing perspective, understanding `quantum execution results` is particularly challenging for developers. As a Qubit can be 0 or 1 with a certain probability, a quantum program that has Qubits as its basic units can have many different states at the same time. The results of a quantum program are certain only when the results are observed (or "measured"). Therefore, it is more challenging for developers to understand the results of quantum programs than that of classical programs. For example "How to plot histogram or Bloch sphere for multiple circuits? I have tried to plot a histogram for the multiple circuits for the code given below. I think I have done it correctly but don't know why it's not working. Please help me out. If possible please solve it for the Bloch sphere" Future efforts are needed to interpret quantum program outputs.

Posts related to `quantum vs. classical computing` are gaining the most attention from developers. Since quantum computing is based on a new philosophy different from classical computing, developers often ask questions about the differences and look to understand the new doors quantum computing is opening. According to Table IV, posts with this topic receive the highest median number of views. This may show that software engineers are eager to contribute to QSE by starting from the differences between the two paradigms. However, as shown in Figure 3, posts on this topic are least likely to receive accepted answers. Our results indicate the need for resources and tools for bridging the knowledge gap between quantum computing and classical computing.

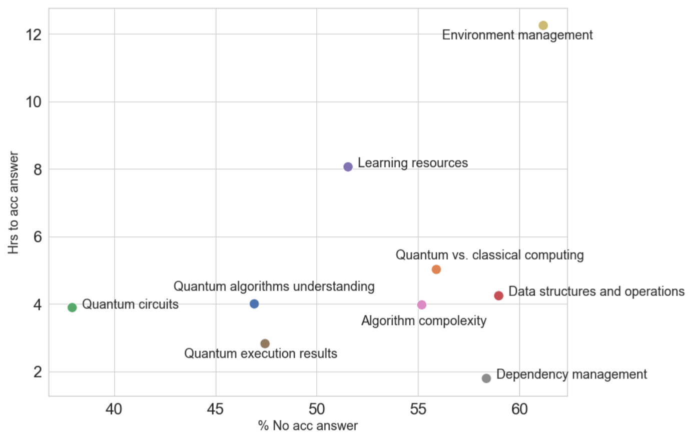

Fig. 3. The difficulty aspect of QSE-related topics on Stack Exchange forums

Questions of some topics (e.g., `environment management`) are much more difficult than others to receive accepted answers. According to Figure 3, the topic `environment management` is the most difficult topic to answer, with 61% of posts not receiving an accepted answer and a median time of 12 hours to receive an accepted answer. `Learning resources`, `quantum vs. classical computing` and `data structures and operations` are also among the most difficult topics in terms of the ratio of posts getting accepted answers and the time to get one. The results indicate the lack of community support in aspects such as setting up a development environment, searching for learning resources, and understanding differences between quantum computing and classical computing, which could impair the advancement of quantum software development practices.

From Q&A forums, we derived nine topics discussed in QSE-related posts, including traditional software engineering topics (e.g., `environment management`) and QSE-specific ones (e.g., `quantum execution results`). We highlighted some particularly challenging areas for QSE, such as interpreting quantum program outputs and bridging the knowledge gap between quantum computing and classical computing.

RQ3: What QSE topics are raised in the issue reports of quantum-computing projects?

*1) **Motivation**:* Issue reports of quantum computing projects record developers' concerns and discussions when adding features or resolving issues in these projects. The textual information in the issue reports may communicate developers' challenges when developing quantum computing applications. Therefore, we analyze the topics in the issue reports to understand the challenges developers are facing as well as the prevalence of these challenges. While the questions on technical forums can provide information about developers' general challenges, the issue reports may communicate developers' challenges for specific problems (i.e., issues).

*2) **Approach**:* **Topic assignment and frequency.** Our topic model on the issue report data generated 17 topics. We follow the same process as described in RQ2 to manually assign meaningful labels to the automated topics, based on the top words in the topics and the content of the associated issue reports. During the manual assignment process, we found that some topics are similar to each other even though such similarity is not detected by the probabilistic topic model. Therefore, we follow prior work [36], [82] and merged similar topics. We also discarded one topic as we could not derive a meaningful label from the top words and the associated issue reports. In the end, we obtained 13 meaningful topics. For each topic, we measure the percentage of the issue reports (i.e., frequency) that have it as the dominant topic (i.e., with the highest probability). We also measure the number and percentage of the studied projects that have at least one issue report of each topic.

Topic difficulty. To further understand developers' challenges in developing quantum computing applications, we measure the "difficulty" of the issue reports associated with each topic. As we cannot directly measure the "difficulty" of issue reports, we measure three indirect metrics for each topic: (1) the percentage of issue reports associated with the topic that is `closed`, (2) the median time required to close an issue (since its creation) associated with the topic, and (3) the median number of comments in an associated topic (intuitively, an issue report with more comments may be more difficult [83]). For each topic, the associated issue reports refer to the issue reports that have it as the dominant topic.

*3) **Results**:* **We derived 13 topics from GitHub issue reports, bringing new perspectives to the challenges faced by QSE developers.** Table V shows the list of our derived topics, their descriptions, their percentage frequency in the studied issue reports, and the number of projects that have at least one issue report of the topic. Among the 13 topics, 6 of them (`learning resources`, `environment management`, `quantum circuits`, `quantum execution results`, `data structures and operations`, and `dependency management`) are overlapping with the topics derived from Stack Exchange posts (RQ2), and another 2 of them (`API change` and `API usage`) are overlapping with the categories of Stack Exchange questions derived in RQ1. This result indicates that the QSE-related challenges that we derived from forum posts indeed impact practical quantum program development in GitHub projects.

Among the other five topics, two of them (i.e., `machine learning` and `quantum chemistry`) are related to the most popular and promising quantum computing application areas: machine learning and chemistry. For example, an issue report associated with `quantum chemistry` raises an issue when using a molecular optimizer Python library: "*it leads to PyBerny optimizing an unconverged ground state energy, which generally leads to the geometry optimization never converging*". The other three topics (i.e., `quantum execution errors`, `unit testing`, `algorithm optimization`) are related to applications of traditional software engineering processes in quantum program development.

All derived topics are general among the quantum computing projects, as each topic is present in the issue reports of 58% to 90% of the projects. We observe that `learning resources` and `environment management` are the two most frequent topics and appear in 90% and 88% of all the studied projects, respectively, which once again highlights the need of efforts for developing learning resources and supporting developers in setting up their quantum program development environment.

Some topics are particularly challenging for developers, such as `data structures and operations`, `quantum circuits`, and `quantum execution results`. Table VI shows the median time it takes to close an issue report and the number of comments in an issue report associated with each topic. All the issues are closed at the time when we analyzed their status. The issues associated with each topic only have a median of one to two comments, indicating that developers' interactions on these issue reports are not intense. `Data structures and operations` is also among the most difficult topics on forum posts (as discussed in RQ2). However, the `Quantum circuits` and `quantum execution results` topics are not among the most difficult topics on forum posts, while they are two of the most difficult ones on GitHub issues, which indicates that quantum circuit issues and the interpretation of quantum program execution results are more difficult in specific problem contexts.

QSE-related challenges that we derived from forum posts indeed impact practical quantum program development in GitHub projects, while GitHub issues bring new perspectives on developers' faced challenges (e.g., on specific quantum computing applications such as machine learning). In particular, we observe that the challenges are generally among the quantum computing projects.

TABLE V
QSE-RELATED TOPICS DERIVED FROM ISSUE REPORTS OF QUANTUM COMPUTING PROJECTS ON GITHUB

Manual label	Keywords	Description	% Freq	# Projects
Learning resources	summary, remove, tutorial, link, documentation	Search for documentation, tutorials, websites, etc.	14.94	109 (90.08%)
Environment management	build, include, library, release, variable	Development environment and build problems	13.72	107 (88.43%)
API change	version, qiskit, code, issue, update	API update or deprecation issues	11.65	98 (80.99%)
Quantum circuits	gate, circuit, qubit, operation, control	Elements of quantum circuits (e.g., Qubits, gates) and their operations	8.30	70 (57.85%)
Quantum chemistry	input, calculation, basis, energy, pyscf	Issues with quantum chemistry libraries (e.g., PySCF)	6.72	76 (62.80%)
Quantum execution errors	error, artiq, follow, experiment, device	Errors in the execution of quantum programs	6.38	80 (66.12%)
Unit testing	test, check, fail, unit, script	Unit testing failures	6.32	87 (71.90%)
API usage	function, method, class, parameter, call	How to use an API	5.81	80 (66.11%)
Quantum execution results	state, number, result, time, measurement	Quantum program execution results (i.e., measured state)	5.76	89 (73.55%)
Data structures and operations	implement, operator, matrix, problem, array	Data structures (e.g., matrix, arrays and list) and their operations	5.73	88 (72.73%)
Machine learning	model, datum, dataset, layer, benchmark	Quantum computing application in machine learning	5.26	75 (61.9%)
Dependency management	file, python, import, package, install	Library installation, use and versioning issues	5.23	93 (76.86%)
Algorithm optimization	case, time, optimization, long, performance	Program performance and algorithm optimization	4.19	83 (68.6%)

TABLE VI
THE DIFFICULTY ASPECT OF QSE-RELATED TOPICS ON GITHUB ISSUES

Topic name	$Hr\ \tilde{to}\ close$	$\#\ com\tilde{m}ents$
Data structures and operations	151.40	1
Quantum circuits	114.98	1
Quantum execution results	98.02	1
Machine learning	94.68	2
API usage	80.70	1
Quantum chemistry	62.89	2
Quantum execution errors	59.44	2
API change	47.34	1
Algorithm optimization	39.83	1
Dependency management	32.40	2
Unit testing	28.26	1
Learning resources	27.02	1
Environment management	21.57	1

All the issues are closed at the time we analyzed their status.

V. THREATS TO VALIDITY

External validity. In this work, we analyze four Stack Exchange forums and 122 GitHub repositories to understand the challenges of QSE. Our studied forum posts and GitHub issues may not cover all the ones that are related to QSE. Developers may also communicate their discussions in other media (e.g., mailing lists). Future work considering other data sources may complement our study. In addition, we identify and collect the posts from Q&A forums using a selected set of tags. Our analysis may miss some QSE tags. However, to alleviate this threat, we follow prior work [60], [61] and use an iterative method to identify the relevant tags.

Internal validity. In this work, we use topic models to cluster the forum posts and GitHub issue reports, based on the intuition that the same clusters would have similar textual information. However, different clusters of posts and issue reports may exist when a different approach is used. To ensure the quality of the clusters, we manually reviewed the resulting topics, merged similar topics when needed, and assigned meaningful labels to them.

Construct validity. In RQ1, we manually analyze the categories of QSE-related questions on technical Q&A forums. Our results may be subjective and depend on the judgment of the researchers who conducted the manual analysis. To mitigate the bias, two authors of the paper collectively conducted an manual analysis and reached a substantial agreement, indicating the reliability of the analysis results. A third author also participated in the discussions during the manual analysis, to ensure the quality of the results. In RQ2 and RQ3, the parameters of the topic models (e.g., the number of topics K) may impact our findings. To mitigate this threat, following previous work [60] [61], we did multiple experiments and use the topic coherence score to select the most suitable parameters. The manual labeling of topics can be subjective. To reduce this threat, the authors read each topic's top 20 keywords and the top 15 highest contributed posts to the topic. We followed a clear-cut approach adapted in previous works [60] [61]. In addition, in our analysis of the QSE posts, we did not filter posts using the number of comments, votes or answers (as done in prior work [84]), which may lead to noise in the analyzed posts (e.g., low-quality posts). We made this decision since QSE is a new topic and the number of posts in the Q&A forums is relatively small.

VI. CONCLUSIONS

This paper examines challenges quantum program developers are facing by analyzing Stack Exchange forums posts related to QSE and the GitHub issue reports of quantum computing projects. Results indicate that QSE developers face traditional software engineering challenges (e.g., dependency management) as well as QSE-specific challenges (e.g., interpreting quantum program execution results). In particular, some QSE-related areas (e.g., bridging the knowledge gap between quantum and classical computing) are gaining the highest attention from developers while being very challenging to them. As the initial effort for understanding QSE-related challenges perceived by developers, our work shed light on future opportunities in QSE (e.g., supporting explanations of theory behind quantum program code and the interpretations of quantum program execution results).

References

[1] W. Knight, "Serious quantum computers are finally here. what are we going to do with them," *MIT Technology Review*, vol. 30, 2018.

[2] D. Maslov, Y. Nam, and J. Kim, "An outlook for quantum computing [point of view]," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 5–10, 2018.

[3] J. Zhao, "Quantum software engineering: Landscapes and horizons," *arXiv preprint arXiv:2007.07047*, 2020.

[4] IBM, "IBM Quantum Computing," https://www.ibm.com/quantum-computing/, Last accessed 05/04/2021.

[5] P. A. M. Dirac, *The principles of quantum mechanics*. Oxford university press, 1981, no. 27.

[6] E. Schrödinger, "Discussion of probability relations between separated systems," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 31, no. 4. Cambridge University Press, 1935, pp. 555–563.

[7] L. Mueck, "Quantum software," *Nature*, vol. 549, no. 171, 2017.

[8] M. Piattini, G. Peterssen, R. Pérez-Castillo, J. L. Hevia, M. A. Serrano, G. Hernández, I. G. R. de Guzmán, C. A. Paradela, M. Polo, E. Murina *et al.*, "The talavera manifesto for quantum software engineering and programming." in *The First International Workshop on the Quantum Software Engineering & Programming*, ser. QANSWER '20, 2020, pp. 1–5.

[9] B. Ömer, "Qcl-a programming language for quantum computers," *Software available on-line at http://tph. tuwien. ac. at/˜ oemer/qcl. html*, 2003.

[10] H. Abraham, AduOffei, R. Agarwal, I. Y. Akhalwaya, and et al., "Qiskit: An open-source framework for quantum computing," 2019.

[11] Google, "Google QuantumAI," https://quantumai.google, Last accessed 05/04/2021.

[12] Microsoft, "Microsoft Azure Quantum Service," https://azure.microsoft.com/en-ca/services/quantum/, Last accessed 05/04/2021.

[13] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[14] G. G. Guerreschi and M. Smelyanskiy, "Practical optimization for hybrid quantum-classical algorithms," *arXiv preprint arXiv:1701.01450*, 2017.

[15] L. O. Mailloux, C. D. Lewis II, C. Riggs, and M. R. Grimaila, "Post-quantum cryptography: what advancements in quantum computing mean for it professionals," *IT Professional*, vol. 18, no. 5, pp. 42–47, 2016.

[16] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, "Elucidating reaction mechanisms on quantum computers," *Proceedings of the National Academy of Sciences*, vol. 114, no. 29, pp. 7555–7560, 2017.

[17] M. Piattini, G. Peterssen, and R. Pérez-Castillo, "Quantum computing: A new software engineering golden age," *ACM SIGSOFT Software Engineering Notes*, vol. 45, no. 3, pp. 12–14, 2020.

[18] E. Moguel, J. Berrocal, J. García-Alonso, and J. M. Murillo, "A roadmap for quantum software engineering: applying the lessons learned from the classics," in *Proceedings of the 1st International Workshop on Software Engineering ++& Technology*, ser. Q-SET '20, 2020.

[19] L. S. Barbosa, "Software engineering for'quantum advantage'," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 427–429.

[20] M. Piattini, M. Serrano, R. Perez-Castillo, G. Petersen, and J. L. Hevia, "Toward a quantum software engineering," *IT Professional*, vol. 23, no. 1, pp. 62–66, 2021.

[21] S. Beyer, C. Macho, M. D. Penta, and M. Pinzger, "What kind of questions do developers ask on stack overflow? a comparison of automated approaches to classify posts into question categories," in *Software Engineering*, 2021.

[22] P. Kaye, R. Laflamme, M. Mosca *et al.*, *An introduction to quantum computing*. Oxford University Press on Demand, 2007.

[23] B. Sodhi, "Quality attributes on quantum computing platforms," *CoRR*, vol. abs/1803.07407, 2018. [Online]. Available: http://arxiv.org/abs/1803.07407

[24] S. Garhwal, M. Ghorani, and A. Ahmad, "Quantum programming language: A systematic review of research topic and top cited languages," *Archives of Computational Methods in Engineering*, pp. 1–22, 2019.

[25] D. Deutsch and R. Penrose, "Quantum theory, the church&#x2013;turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1985.0070

[26] E. Knill, "Conventions for quantum pseudocode," 6 1996. [Online]. Available: https://www.osti.gov/biblio/366453

[27] J. W. Sanders and P. Zuliani, "Quantum programming," in *Mathematics of Program Construction*, ser. Lecture Notes in Computer Science, vol. 1837. Springer, 2000, pp. 80–99.

[28] H. MLNAŘÍK, "Semantics of quantum programming language lanq," *International Journal of Quantum Information*, vol. 06, no. supp01, pp. 733–738, 2008. [Online]. Available: https://doi.org/10.1142/S0219749908004031

[29] K. Svore, A. Geller, M. Troyer, J. Azariah, C. Granade, B. Heim, V. Kliuchnikov, M. Mykhailova, A. Paz, and M. Roetteler, "Q#: Enabling scalable quantum computing and development with a high-level dsl," in *Proceedings of the Real World Domain Specific Languages Workshop 2018*, ser. RWDSL2018. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3183895.3183901

[30] R. LaRose, "Overview and comparison of gate level quantum software platforms," *Quantum*, vol. 3, p. 130, 2019.

[31] B. Vasilescu, "Academic papers using stack exchange data," https://meta.stackexchange.com/questions/134495/academic-papers-using-stack-exchange-data, Last accessed 05/04/2021.

[32] S. Wang, D. Lo, and L. Jiang, "An empirical study on developer interactions in stackoverflow," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1019–1024.

[33] H. Chen, J. Coogle, and K. Damevski, "Modeling stack overflow tags and topics as a hierarchy of concepts," *Journal of Systems and Software*, vol. 156, pp. 283–299, 2019.

[34] M. Allamanis and C. Sutton, "Why, when, and what: analyzing stack overflow questions by topic, type, and code," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 53–56.

[35] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using stack overflow," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 93–96.

[36] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[37] P. K. Venkatesh, S. Wang, F. Zhang, Y. Zou, and A. E. Hassan, "What do client developers concern when using web apis? an empirical study on developer forums and stack overflow," in *2016 IEEE International Conference on Web Services (ICWS)*. IEEE, 2016, pp. 131–138.

[38] M. Alshangiti, H. Sapkota, P. K. Murukannaiah, X. Liu, and Q. Yu, "Why is developing machine learning applications challenging? a study on stack overflow posts," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2019, pp. 1–11.

[39] S. Ahmed and M. Bagherzadeh, "What do concurrency developers ask about? a large-scale study using stack overflow," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–10.

[40] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016.

[41] J. Zou, L. Xu, M. Yang, X. Zhang, and D. Yang, "Towards comprehending the non-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis," *Information and Software Technology*, vol. 84, pp. 19–32, 2017.

[42] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, and X. Zhang, "Which non-functional requirements do developers focus on? an empirical study on stack overflow using topic analysis," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 446–449.

[43] Y. Zhang, D. Lo, X. Xia, and J.-L. Sun, "Multi-factor duplicate question detection in stack overflow," *Journal of Computer Science and Technology*, vol. 30, no. 5, pp. 981–997, 2015.

[44] C. Treude and M. Wagner, "Predicting good configurations for github and stack overflow topic models," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 84–95.

[45] T. Zhang, G. Yang, B. Lee, and E. K. Lua, "A novel developer ranking algorithm for automatic bug triage using topic model and developer relations," in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 1. IEEE, 2014, pp. 223–230.

[46] X. Xia, D. Lo, X. Wang, and B. Zhou, "Accurate developer recommendation for bug resolution," in *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013, pp. 72–81.

[47] H. Naguib, N. Narayan, B. Brügge, and D. Helal, "Bug report assignee recommendation using activity profiles," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 22–30.

[48] X. Xia, D. Lo, Y. Ding, J. M. Al-Kofahi, T. N. Nguyen, and X. Wang, "Improving automated bug triaging with specialized topic model," *IEEE Transactions on Software Engineering*, vol. 43, no. 3, pp. 272–297, 2016.

[49] A. Hindle, A. Alipour, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection and ranking," *Empirical Software Engineering*, vol. 21, no. 2, pp. 368–410, 2016.

[50] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun, "Duplicate bug report detection with a combination of information retrieval and topic modeling," in *2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2012, pp. 70–79.

[51] J. Zou, L. Xu, M. Yang, M. Yan, D. Yang, and X. Zhang, "Duplication detection for software bug reports based on topic model," in *2016 9th International Conference on Service Science (ICSS)*. IEEE, 2016, pp. 60–65.

[52] A. T. Nguyen, T. T. Nguyen, J. Al-Kofahi, H. V. Nguyen, and T. N. Nguyen, "A topic-based approach for narrowing the search space of buggy files from a bug report," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, 2011, pp. 263–272.

[53] L. Martie, V. K. Palepu, H. Sajnani, and C. Lopes, "Trendy bugs: Topic trends in the android bug reports," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 2012, pp. 120–123.

[54] A. Aggarwal, G. Waghmare, and A. Sureka, "Mining issue tracking systems using topic models for trend analysis, corpus exploration, and understanding evolution," in *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, 2014, pp. 52–58.

[55] "Stack Overflow forum," https://stackoverflow.com, Last accessed 05/04/2021.

[56] "Stack Exchange Quantum Computing forum," https://quantumcomputing.stackexchange.com, Last accessed 05/04/2021.

[57] "Stack Exchange Computer Science forum," https://cs.stackexchange.com, Last accessed 05/04/2021.

[58] "Stack Exchange Artificial Intelligence forum," https://ai.stackexchange.com, Last accessed 05/04/2021.

[59] "stack exchange data explorer."

[60] G. Uddin, O. Baysal, L. Guerrouj, and F. Khomh, "Understanding how and why developers seek and analyze api-related opinions," 2021.

[61] M. Openja, B. Adams, and F. Khomh, "Analysis of modern release engineering topics : – a large-scale study using stackoverflow –," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2020, pp. 104–114.

[62] B. Cartaxo, G. Pinto, D. Ribeiro, F. K. Kamei, R. Santos, F. Silva, and S. Soares, "Using q&a websites as a method for assessing systematic reviews," 05 2017.

[63] J. Businge, M. Openja, S. Nadi, E. Bainomugisha, and T. Berger, "Clone-based variability management in the android ecosystem," *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 625–634, 2018.

[64] J. Businge, M. Openja, D. Kavaler, E. Bainomugisha, F. Khomh, and V. Filkov, "Studying android app popularity by cross-linking github and google play store," *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 287–297, 2019.

[65] "GitHub REST API," https://developer.github.com/v3/, Last accessed 05/04/2021.

[66] P. Willett, "The porter stemming algorithm: Then and now," *Program electronic library and information systems*, vol. 40, 07 2006.

[67] V. Gurusamy and S. Kannan, "Performance analysis: Stemming algorithm for the english language," *International Journal for Scientific Research and Development*, vol. 5, pp. 2321–613, 08 2017.

[68] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[69] T.-H. Chen, S. W. Thomas, and A. E. Hassan, "A survey on the use of topic models when mining software repositories," *Empirical Software Engineering*, vol. 21, no. 5, pp. 1843–1919, 2016.

[70] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[71] R. Rossi and S. Rezende, "Generating features from textual documents through association rules," 01 2011.

[72] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002, http://mallet.cs.umass.edu.

[73] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, 2015.

[74] "Gensim coherencemodel implimentation," https://radimrehurek.com/gensim/models/coherencemodel.html, Last accessed 05/04/2021.

[75] J. Han, E. Shihab, Z. Wan, S. Deng, and X. Xia, "What do programmers discuss about deep learning frameworks," *Empirical Software Engineering*, vol. 25, pp. 2694–2747, 2020.

[76] S. Beyer, C. Macho, M. Di Penta, and M. Pinzger, "What kind of questions do developers ask on stack overflow? a comparison of automated approaches to classify posts into question categories," *Empirical Software Engineering*, vol. 25, no. 3, pp. 2258–2301, 2020.

[77] H. Li, W. Shang, B. Adams, M. Sayagh, and A. E. Hassan, "A qualitative study of the benefits and costs of logging from developers' perspectives," *IEEE Transactions on Software Engineering*, 2020.

[78] M. McHugh, "Interrater reliability: The kappa statistic," *Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB*, vol. 22, pp. 276–82, 10 2012.

[79] "Cirq," Mar. 2021, See full list of authors on Github: https://github .com/quantumlib/Cirq/graphs/contributors. [Online]. Available: https://doi.org/10.5281/zenodo.4586899

[80] X.-L. Yang, D. Lo, X. Xia, Z. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, pp. 910–924, 09 2016.

[81] M. Bagherzadeh and R. Khatchadourian, "Going big: A large-scale study on what big data developers ask," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 432–442. [Online]. Available: https://doi.org/10.1145/3338906.3338939

[82] E. Noei, F. Zhang, and Y. Zou, "Too many user-reviews, what should app developers look at first?" *IEEE Transactions on Software Engineering*, 2019.

[83] F. Khomh, B. Chan, Y. Zou, and A. E. Hassan, "An entropy evaluation approach for triaging field crashes: A case study of mozilla firefox," in *2011 18th Working Conference on Reverse Engineering*, 2011, pp. 261–270.

[84] E. Farhana, N. Imtiaz, and A. Rahman, "Synthesizing program execution time discrepancies in julia used for scientific software," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2019, pp. 496–500.